

Diagramming in Software Engineering

Álvaro García Miranda
Marcos Machado Menéndez
Óscar Abad López

¿Qué es la diagramación?

- ▶ Lo primero es entender qué es la diagramación. La diagramación es el oficio de diseño editorial que se encarga de organizar en un espacio, escrito, visual y en algunos casos contenido audiovisual en medios impresos y electrónicos.

UML

UML viene de las siglas en inglés de lenguaje unificado de modelado, nos permite representar visualmente la arquitectura, diseño e implementación de sistemas de software complejos.

Algunas ventajas:

1. Explicar el código a gente nueva en el proyecto
2. Navegar por el código fuente
3. Planificar nuevas características antes de programar
4. Comunicarse con grupos de técnicos y de personas que no lo son de forma simultánea

Tipos de Diagramas UML

- ▶ Los podemos clasificar en dos tipos en función de la información que nos proporciona cada diagrama:
 1. Estructurales
 2. De comportamiento

Diagramas Estructurales

- ▶ Muestran la organización del sistema:
 1. Diagrama de Clases
 2. Diagrama de Componentes
 3. Diagrama de Implementación
 4. Diagrama de Estructura Compuesta
 5. Diagrama de Objetos
 6. Diagrama de Paquetes

Diagramas de Comportamiento

- ▶ Muestran la forma en la que se comporta el sistema y cómo interactúa dentro de sí mismo, con los usuarios, otros sistemas y otras entidades:
 1. Diagrama de Temporización
 2. Diagrama Global de Interacciones
 3. Diagrama de Comunicación o de Colaboración
 4. Diagrama de Estados
 5. Diagrama de Casos de Usos
 6. Diagrama de Secuencia
 7. Diagrama de Actividades

Herramientas para crear diagramas

- ▶ Tenemos dos opciones a la hora de crear diagramas:
 1. Hacerlo de forma manual, ya sea en papel o mediante herramientas digitales.
 2. Hacerlo de forma automática

Herramientas Manuales

- ▶ Presentan varios problemas a la hora de mantener actualizada la documentación.
- ▶ Hay algunos casos en que son muy útiles, pero no deberíamos depender de herramientas manuales para crear todos los diagramas de nuestra arquitectura.

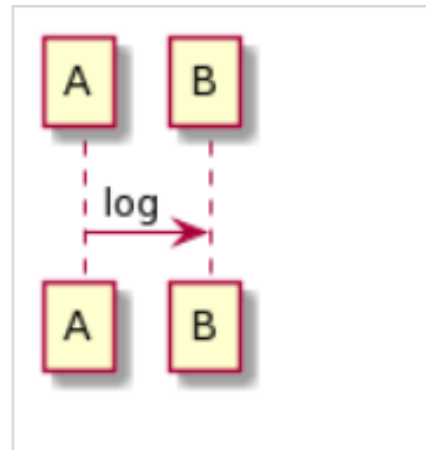
Herramientas Automáticas

- ▶ Las herramientas automáticas nos proporcionan una solución a los problemas de mantenibilidad que nos presentaban las herramientas manuales. Es más sencillo que nuestros diagramas estén actualizados y no perdamos información.
- ▶ De nuevo, hay varias opciones que debemos contemplar:
 1. PlantUML
 2. Mermaid

PlantUML

- ▶ PlantUML es una herramienta que emplea lenguaje Markup:

```
1 @startuml
2
3 skin rose
4
5 A -> B: log
6
7 @enduml
```



- ▶ Está hecha con java, lo que impide su uso por parte de algunos usuarios.

Mermaid

- ▶ Es una herramienta que comparte con PlantUML el utilizar lenguaje Markup lo que hace que la sintaxis a usar es prácticamente la misma.
- ▶ Sin embargo, está hecha con JavaScript, lo que permite que se ejecute en navegador. Esto hace que la información sea aún más accesible.
- ▶ Está soportado de forma nativa por GitHub.

Modelo C4

- ▶ Creado por Simon Brown, consiste en un conjunto jerárquico de diagramas de arquitectura de software.
- ▶ Al estar organizada la información en una jerarquía permite un nivel de abstracción para cada capa en función de las necesidades, ya que cada nivel de la jerarquía está destinada a un público concreto.
- ▶ Organiza la información en cuatro niveles:
 1. Contexto
 2. Contenedores
 3. Componentes
 4. Código

Diagrama de Contexto

- Muestra el sistema de software que está construyendo y cómo encaja en el mundo en términos de las personas que lo utilizan y los otros sistemas de software con los que interactúa

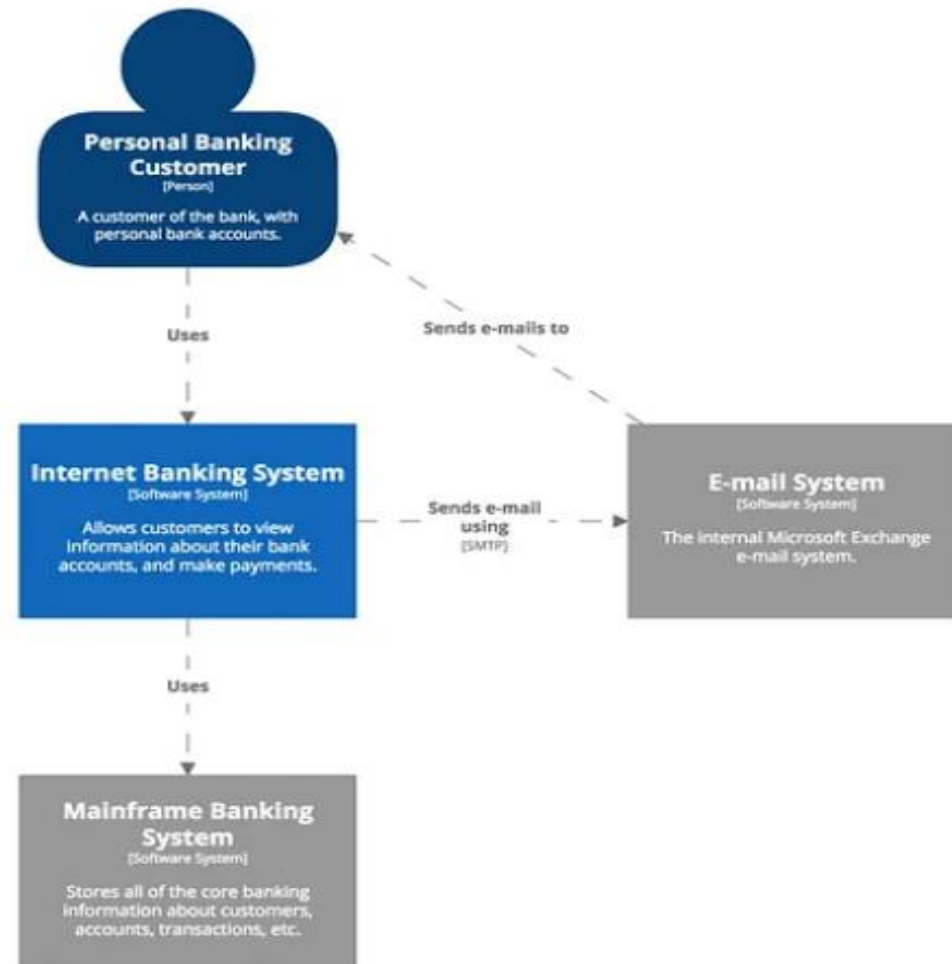


Diagrama Contenedor

- Amplía el sistema de software y muestra los contenedores (aplicaciones, almacenamiento de datos, microservicios, etc.) que componen este sistema de software.

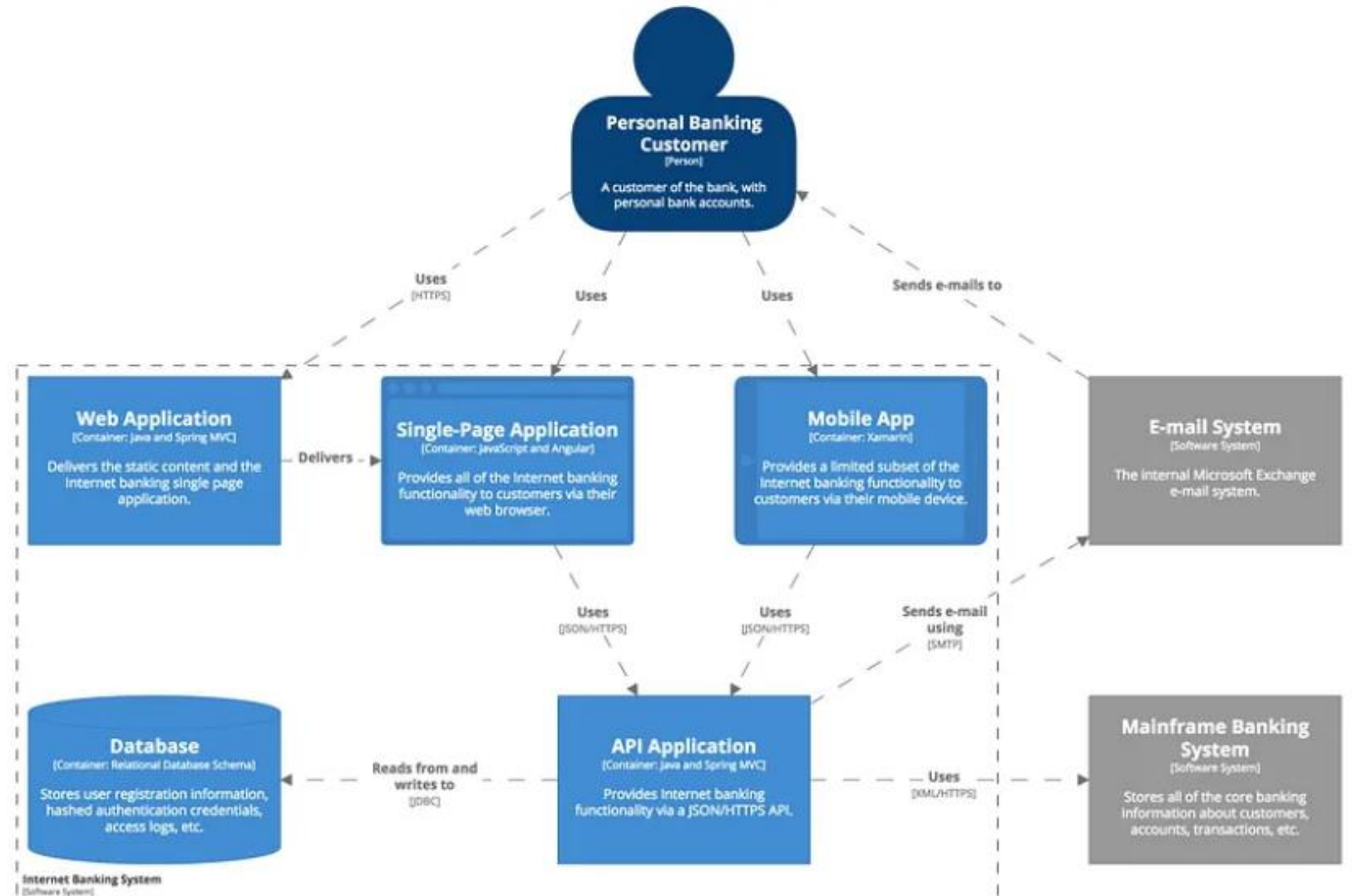
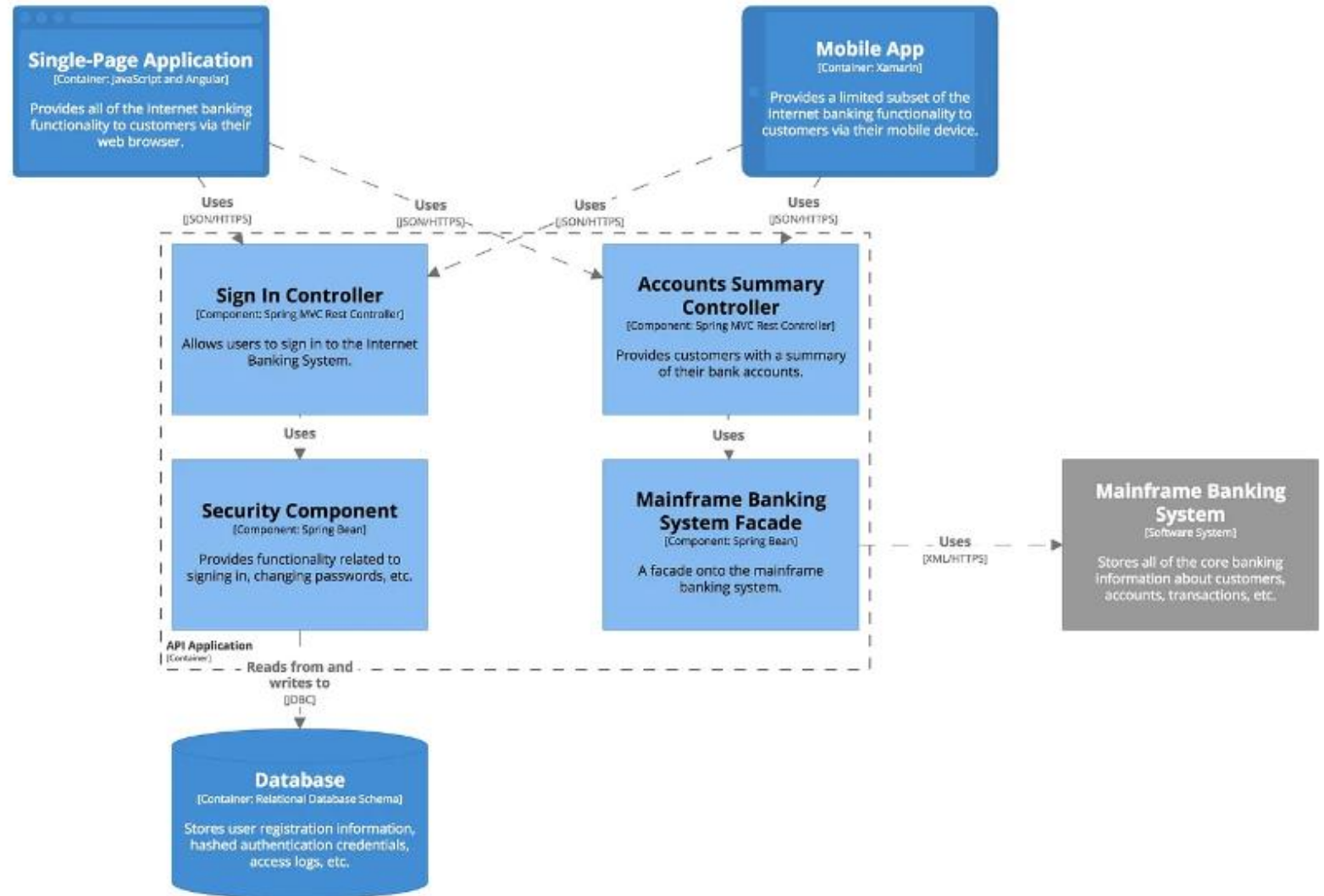


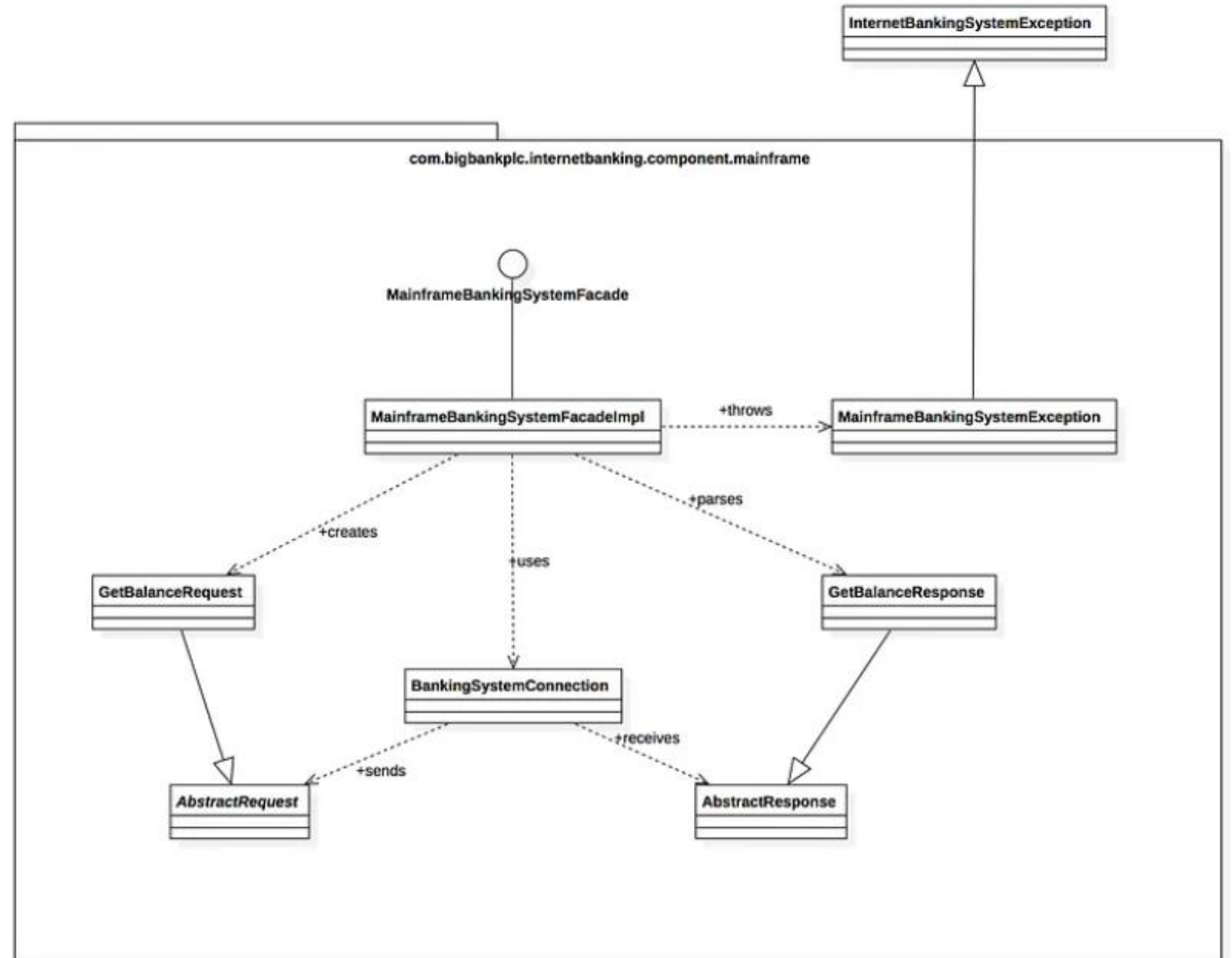
Diagrama de Componentes

- ▶ Expande un contenedor individual para mostrar los componentes que contiene. Estos componentes deben asignarse a abstracciones reales (por ejemplo, una agrupación de códigos) en función de su código



Código

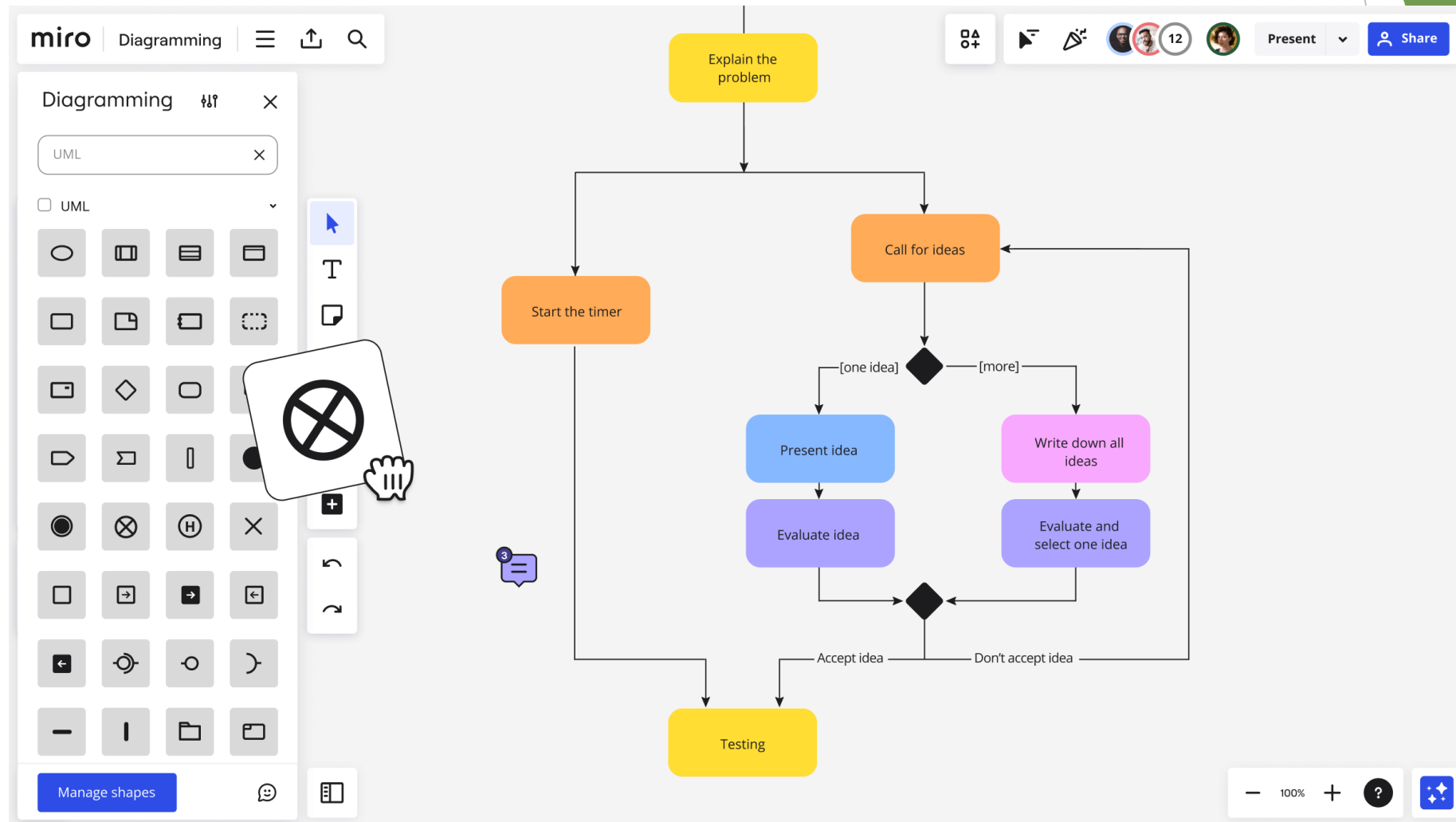
- Se puede ampliar un componente individual para mostrar cómo se implementa este componente



Whiteboarding

- ▶ Técnica más informal para realizar diagramas
- ▶ Similar al brainstorming
- ▶ Un grupo puede ver y editar una pizarra común
- ▶ Menos mantenible
- ▶ Herramienta recomendada: Miro

Whiteboarding



Co-generation

- ▶ Consiste en generar código a partir de diagramas
- ▶ No está muy avanzado
- ▶ Crea un esqueleto básico, pero sin funcionalidad

Co-generation

The screenshot displays the Software Ideas Modeler Ultimate interface. The main workspace shows a UML class diagram with the following elements:

- Category Class:** Attributes include ID (Integer {id}), Name (String), Description (String), ParentCategory (Category), and CreationDate (DateTime). It has a self-association labeled "-subCategories" with multiplicity "0..1" and a composition relationship with the Application class labeled "-parentCategory".
- Publisher Class:** Attributes include ID (Integer {id}), Name (String), Description (String), and CreationDate (DateTime). It is associated with the Application class.
- Application Class:** Attributes include ID (Integer {id}), Name (String), Description (String), Size (Integer), PublishedDate (DateTime), Status (ApplicationStatus = Filed), and DownloadCounter (Integer). It is associated with the ApplicationStatus enumeration.
- ApplicationStatus Enumeration:** Values include Draft, Filed, Published, Rejected, Hidden, Deleted, and Approved.

The right-hand pane shows the generated C# code:

```
Generate From
Whole Diagram Selected Elements

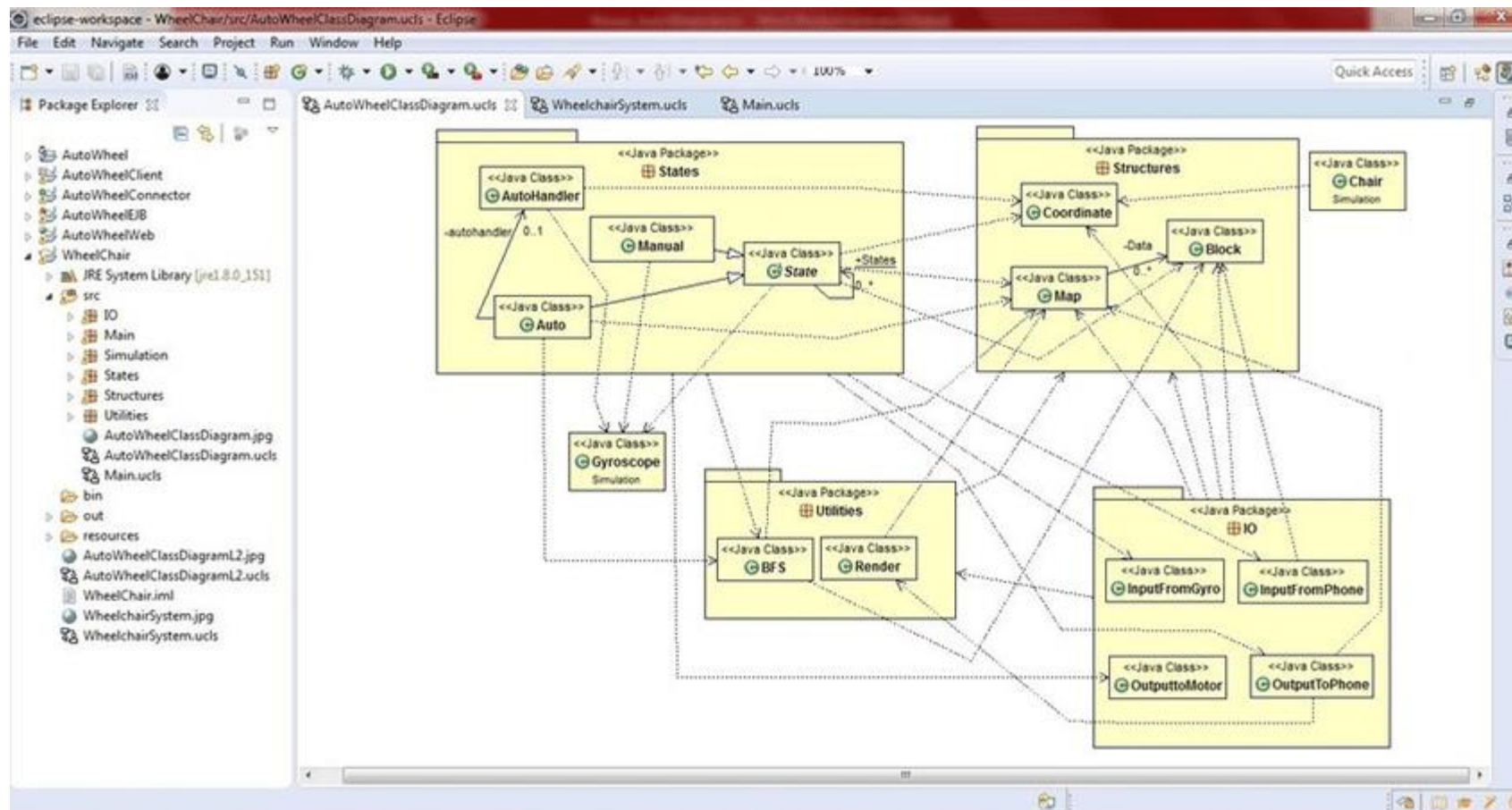
//Publisher of applications.
internal class Publisher
{
    private int ID;
    private string Name;
    private string Description;
    private DateTime CreationDate;
}

namespace StoreModel
{
    //The status of the application processing
    internal enum ApplicationStatus
    {
        Draft,
        Filed,
        Published,
        Rejected,
        Hidden,
        Deleted,
        Approved
    }
}
```

The status bar at the bottom indicates: X: 622px; Y: 401px; 1 item(s) selected; Offline; Registered Copy; © 2009 - 2020 Dusan Rodina; Version: 12.64; 100%.

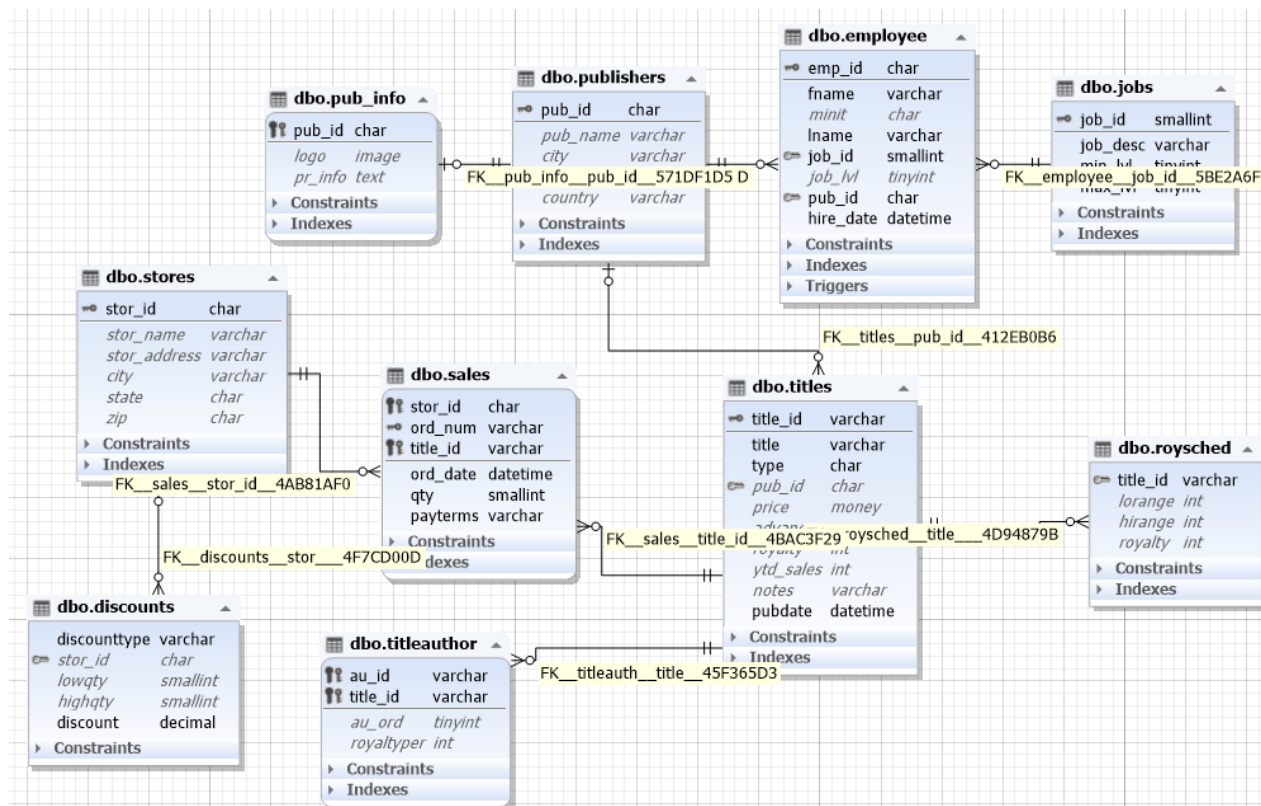
Generar diagramas a partir de código

- Muchos IDEs permiten generar diagramas de clases a partir del código



Generar diagramas a partir de código

- ▶ Los IDEs de bases de datos también permiten generar diagramas de entidad-relación



Cómo facilitar la entrada en un proyecto

- ▶ Entrar en un proyecto sin conocer el dominio es difícil
- ▶ Se recomienda:
 - Tener un modelo de dominio
 - Tener un modelo C4 de alto nivel
- ▶ Se pueden complementar con explicaciones escritas
- ▶ Mantener la documentación actualizada
- ▶ No se recomiendan los diagramas short-lived